VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ONLINE NABÍDKOVÝ SYSTÉM

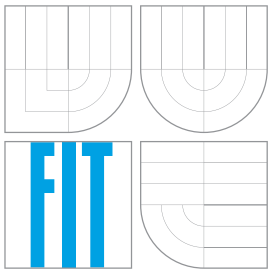
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAROSLAV MOLTAŠ

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ONLINE NABÍDKOVÝ SYSTÉM

ONLINE BIDDING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAROSLAV MOLTAŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LADISLAV RUTTKAY

BRNO 2010

Abstrakt

Cílem této práce je vytvořit webový informační systém, který umožní firemní evidenci produktů, služeb, zákazníku. Systém umožní přihlášení obchodního zástupce, vytvoření nabídky pro zákazníka, následně umožní evidenci nabídky, tisk nebo odeslání elektronickou poštou.

Abstract

The goal of this thesis is to create web information system, which can file products, services and customers of the company. System enables to login of sales representative, to create bid for customer, then enables to file bid, print it or send via electronic mail.

Klíčová slova

Nabídkový systém, Webová aplikace, Cenová nabídka, .NET Framework, ASP.NET, C#, SQL, OOP

Keywords

Bidding System, Web Application, Price Quotation, .NET Framework, ASP.NET, C#, SQL, OOP

Citace

Jaroslav Moltaš: Online nabídkový systém, bakalářská práce, Brno, FIT VUT v Brně, 2010

Online nabídkový systém

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Ladislava Ruttkaye

.....
Jaroslav Moltaš
16. května 2010

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době se čím dál více setkáváme s trendy přesunu aplikací na web a vytváření tzv. webových aplikací. Výhodou takovýchto aplikací je především snížení nákladů na vlastnictví, žádná instalace a dostupnost. Je možné se k nim připojit prakticky odkudkoli na světě, pokud je dostupné připojení k internetu. K tomuto účelu je vhodná technologie ASP.NET od Microsoftu.

Mnoho firem tímto způsobem prezentuje své nabídky, e-shopy a aplikace. Další možností využití webových aplikací jsou různé informační či interní systémy. Tímto druhem webové aplikace se budeme zabývat.

Cílem této práce je vytvořit webový informační systém, který umožní firemní evidenci produktů, služeb, zákazníku. Systém umožní přihlášení obchodního zástupce, vytvoření nabídky pro zákazníka, následně umožní evidenci nabídky, tisk nebo odeslání elektronickou poštou.

Schopnosti a funkčnost systému budou předvedeny na vzorové firmě, která je zaměřena na výrobu a prodej svařovací techniky. Tato firma bude nabízet několik hlavních produktů (svářečky) a k nim kompatibilní zařízení (hořáky, zdroje), jiná příslušenství (oděv, spreje, kapaliny) a několik služeb (montáž, uvedení do provozu, zaškolení).

Kapitola 2

Použité technologie

Jak bylo řečeno v úvodu studie, informační systém je založen zejména na technologii od společnosti Microsoft, implementačním jazyku .NET Framework.

2.1 Microsoft .NET Framework

.NET Framework je platforma pro vytváření a provozování aplikací. Microsoft .NET je převážně o webových službách XML, ale .NET Framework podporuje i další programovací modely. Kromě zápisu webových služeb můžete psát konzolové aplikace, aplikace GUI („formuláře Windows“), webové aplikace („webové formuláře“) a dokonce i služby Windows, častěji označovány za služby NT. Celý rámec nám také pomáhá *konzumovat* webové služby – tedy zapisovat klienty webových služeb. Aplikace vytvořené v prostředí .NET Framework však nemusejí webové služby používat [10].

Vedle webových služeb XML je další součástí rámce s největším potenciálem změnit svět ASP.NET.

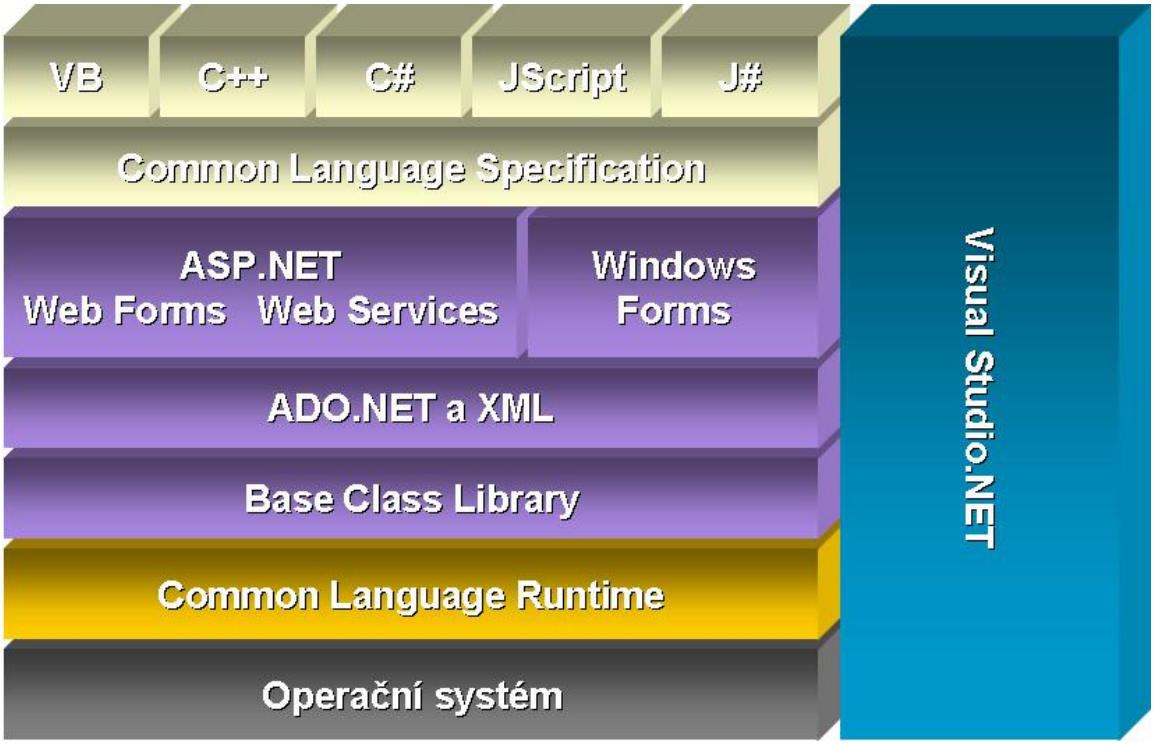
2.1.1 Architektura .NET Framework

Následující odstavec je převzat z [3].

Na nejnižší úrovni se nachází CLR – Common Language Runtime realizující základní infrastrukturu, nad kterou je framework vybudován. Nad CLR se nachází několik hierarchicky umístěných knihoven. Ty jsou rozděleny do jmenných prostorů. Základem je knihovna nazvaná Base Class Libary. Nad ní je knihovna pro přístup k datům a práci s XML soubory. Poslední vrstvou je sada knihoven usnadňující práci s uživatelským rozhraním. Je rozdělena do dvou skupin: pro usnadnění vytváření webových aplikací a pro vytváření klasických aplikací. Poslední vrstvu tvoří nelimitovaná množina programovacích jazyků. Jejich základní vlastnosti definuje CLS – Common Language Specification. V současné době jsou firmou Microsoft podporovány čtyři jazyky: Visual Basic, C++, C# a Jscript. Tato množina ale není uzavřena a jakýkoliv výrobce ji může rozšířit.

2.1.2 ASP.NET

ASP.NET je součást platformy .NET Framework společnosti Microsoft. Tento název pochází z aktivních serverových stránek (Active Server Pages – ASP), které revolucionalizovaly webové programování v 90. letech tím, že nabídly snadno použitelný model dynamického vytváření obsahu HTML na webových serverech pomocí serverových skriptů. ASP.NET



Obrázek 2.1: Architektura .NET Framework (převzato z [3])

je další verzí ASP, jež poskytuje lákavý nový způsob psaní webových aplikací, který se nepodobá ničemu zatím existujícímu [10].

Tato technologie přistupuje k tvorbě webových aplikací s využitím webových formulářů. Stránky jsou tvořeny ovládacími prvky a těmto prvkům je možné přiřazovat vlastnosti, zachytávat události atp.

2.2 Microsoft SQL Server

Microsoft SQL Server je relační databázový systém programovaný firmou Microsoft. Podporuje standardy jazyka SQL (Structured Query Language (strukturovaný dotazovací jazyk)) a zavádí jeho další rozšíření Transact-SQL (T-SQL). V současné době je nejnovější verze SQL Server 2008.

2.3 XHTML

HTML (HyperText Markup Language) je standardní jazyk pro tvorbu hypertextových dokumentů. Je jedním z jazyků pro vytváření WWW stránek.

XHTML je novější norma jazyka HTML. X na začátku znamená eXtensible - rozšiřitelný (ve skutečnost jde o zúžení a osekání). Jde o reformulaci jazyka HTML jako aplikaci XML a má tedy přísnější pravidla. Např. všechny názvy značek (tagů) a atributů musí být malými písmeny, všechny prvky musí být uzavřeny (i nepárové značky), hodnoty atributů musí být uzavřeny do uvozovek atd.

Výhody XHTML oproti HTML jsou především [9]:

- Díky přísným a zároveň jednoduchým pravidlům, mohou počítače XML a tedy i XHTML velmi snadno automatizovaně zpracovávat. Kdyby prohlížeči stačilo „umět“ XHTML, byl by mnohem jednodušší (a tedy menší a rychlejší), než když musí zvládat veškeré „nevyzpytatelnosti“ HTML.
- Všechny aplikace XML mohou s výhodou těžit ze stejného základu syntaktických pravidel. Již nyní tedy existuje mnoho univerzálních programů a knihoven funkcí, které velmi usnadňují vznik a implementaci každé nové aplikace XML.
- Dá se očekávat, že právě díky vyšší srozumitelnosti počítačům budou časem stránky vytvořené v XHTML „oblíbenější“ u vyhledávačů, katalogů stránek, výměnných reklamních systémů a dalších automatizovaných služeb.
- Dříve nebo později začnou prohlížeče podporovat pouze XHTML (případně jiné aplikace XML).

2.4 CSS

CSS (Cascading Style Sheets) neboli kaskádové styly slouží k popisu vzhledu stránek (nezávisí přitom na obsahu) napsaných v jazycích HTML, XHTML nebo XML.

2.5 XML

XML (Extensible Markup Language) je značkovací jazyk vycházející z jazyka SGML. **Výhody** XML [13]:

- *Standardní formát pro výměnu informací*
  Tento formát není svázán s žádnou platformou nebo technologií a je zpracovatelný libovolným textovým editorem.
- *Mezinárodní podpora*
  V XML můžeme vytvářet dokumenty, které obsahují texty v mnoha jazycích najednou – můžeme přepínat mezi různými jazyky v jednom dokumentu. Současně je přípustné i jiné libovolné kódování (např. windows-1250, iso-8859-2), musí však být v každém dokumentu přesně určeno.
- *Vysoký informační obsah*
  Pomocí XML značek (tagů) vyznačujeme v dokumentu význam jednotlivých částí textu. Dokumenty tak obsahují více informací, než kdyby se používalo značkování zaměřené na prezentaci (vzhled) – definice písma, odsazení a podobně. XML dokumenty jsou informačně bohatší.
- *Snadná konverze do jiných formátů*
  Při používání XML dokumentu potřebujeme také dokument zobrazit. XML samo o sobě žádné prostředky pro definici vzhledu nenabízí. Existuje ale několik stylových jazyků, které umožňují definovat, jak se mají jednotlivé elementy zobrazit. Souboru pravidel nebo příkazů, které definují, jak se dokument převede do jiného formátu, se říká styl. Jeden vytvořený styl můžeme aplikovat na mnoho dokumentů stejného typu, stejně tak můžeme na jeden dokument aplikovat několik různých stylů. Výsledkem může být např. PostScriptový soubor, HTML kód nebo XML s obsahem původního dokumentu.

- *Automatická kontrola struktury dokumentu*
  XML neobsahuje předdefinované značky (tagy), je třeba definovat vlastní značky, které budeme používat. Tyto značky je možné (nepovinně) definovat v souboru DTD (Document Type Definition). Potom je možné automaticky kontrolovat, zda vytvářený XML dokument odpovídá této definici. Program, který tyto kontroly provádí, se nazývá parser. Při vývoji aplikací můžeme parser použít, a ten za nás detekuje většinu chyb v datech.

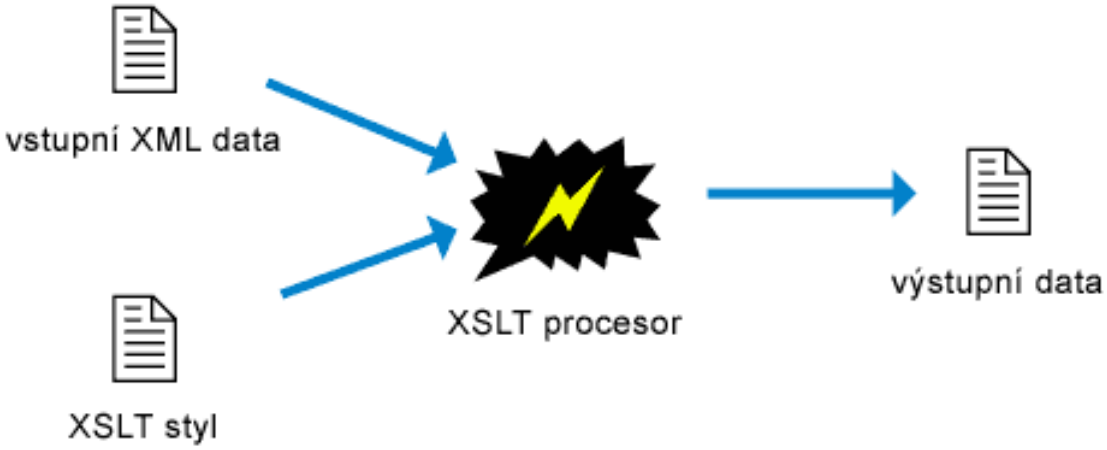
- *Hypertext a odkazy*
  XML stejně jako HTML umožňuje vytváření odkazů v rámci jednoho dokumentu i mezi dokumenty, má však více možností. Je možné vytvářet i vícesměrné odkazy, které spojují více dokumentů dohromady. Tvorba odkazů je popsána ve třech standardech – XLink, XPointer a XPath.

2.6 XSL

Specifikace XSL v sobě zahrnuje dvě části: XSLT a XSL-FO.

2.6.1 XSLT

XSLT (eXtensible Stylesheet Language Transformations) slouží k manipulaci dat ve formátu XML. Je možné tyto soubory transformovat do jiného požadovaného formátu (HTML, jiného XML, PDF,..).



Obrázek 2.2: Transformace XML (převzato z [4])

2.6.2 XSL-FO

Tato technologie slouží k formátování vzhledu XML souborů.

Kapitola 3

Použité nástroje

V této kapitole se budeme věnovat nástrojům a programům, pomocí kterých byla práce tvořena.

3.1 Microsoft Visual Studio 2008

Následující odstavec je převzat z [6].

Visual Studio 2008 poskytuje pokročilé vývojové nástroje, ladicí funkce, funkce pro práci s databázemi a vynalézavé novinky pro rychlou tvorbu špičkových aplikací různých typů. Přináší četná vylepšení: vizuální návrháře pro rychlejší vývoj s využitím .NET Framework 3.5, podstatná zdokonalení webových vývojových nástrojů a vylepšení jazyka, která zrychlují vývoj pro data všech typů. Visual Studio 2008 nabízí vývojářům všechny nástroje a podporu frameworku, které jsou zapotřebí k vytváření skvělých webových aplikací využívajících technologii AJAX. Díky vysoce funkčním frameworkům na straně klienta i serveru budou vývojáři schopni snadno budovat klientské webové aplikace, integrované s libovolným datovým úložištěm, běžící v kterémkoliv moderním prohlížeči a s plným přístupem k aplikačním službám ASP.NET a k platformě Microsoft.

3.2 Rise Editor

Rise Editor je softwarová sada pro vytváření řízených informačních systémů [11]. Pomocí tohoto nástroje lze jednoduše vytvořit ER-diagram a výsledek exportovat do různých typů souborů.

3.3 Violet UML Editor

Violet UML Editor [1] je nástroj pro snadnou tvorbu a editaci Use-case diagramů. Diagramy je možné exportovat jako obrázky.

Kapitola 4

Analýza a návrh

V analýze se budeme zabývat rozborem databázového modelu a specifikací požadavků na systém. Databázový model znázorníme pomocí ER-diagramu a na požadavky na systém využijeme Use Case model.

4.1 Uživatelské účty

ASP.NET obsahuje model správy uživatelských účtů a oprávnění. Ten se skládá ze tří částí [12]:

- *Membership*
  Tato část se stará o samotné uživatelské účty, jména a hesla, jejich vytváření, změny a mazání. Stará se tedy o identifikaci a autentizaci uživatelů.
- *Roles*
  Každého uživatele z Membershipu můžeme přiřadit do několika různých rolí, podle nichž pak provádíme autorizaci. Každý uživatel může být ve více rolích a každá role může obsahovat víc uživatelů.
- *Profiles*
  O každém uživateli si můžeme pamatovat mnoho různých dalších údajů (jméno, příjmení, adresu, telefon, ale např. i nastavení a přizpůsobení webu jeho požadavkům, případně u e-shopů např. nákupní košík apod.).

ASP.NET obsahuje vestavěné providery – *SqlMembershipProvider*, *SqlRolesProvider* a *SqlProfileProvider*. Ty si standardně vytváří vlastní databázi v adresáři **App_Data** a uživatelské údaje ukládají tam. Tabulková struktura, kterou používají, je ale zbytečně komplikovaná a v našem případě nevhodná.

Proto v naší aplikaci použijeme sadu providerů **AltairisWebProviders** volně dostupné na [5]. Výhodou je jejich databázová reprezentace a možnost provázání s naší databázovou strukturou.

4.2 Entity-relationship model

Entity-relationship model (ERM) se používá pro abstraktní a konceptuální zobrazení dat. Diagramy vytvořené pomocí této metody se nazývají entity-relationship diagramy, ER diagramy (ERD). Konceptuální model je později převeden na relační model, který slouží k vytvoření databáze.

4.2.1 Datové typy modelující jednotlivé informace

V našem návrhu jsou zobrazeny i entity představující uživatele, které vyplývají z použití providerů popsaných v předchozí části.

Users – entita představuje uživatele, který se může přihlásit do systému a podle role vykonávat činnosti. O každého uživatele budeme evidovat:

- identifikační číslo,
- uživatelské jméno,
- heslo tvořené jako hash a salt,
- e-mail,
- komentář,
- datum, kdy byl uživatel vytvořen,
- datum, kdy byl uživatel naposledy přihlášen,
- datum, kdy uživatel naposledy měnil heslo.

Roles – tato entita obsahuje seznam rolí, ke kterým může být uživatel přidělen. Budeme tedy uchovávat pouze její název.

Profiles – entita představuje profil každého uživatele, o kterém budeme uchovávat:

- jméno uživatele skládající se z titulu, jména a příjmení,
- kontaktní adresa skládající se z města, ulice a poštovního směrovacího čísla,
- telefon,
- fax,
- datum narození.

Customers – zákazník, pro kterého bude vytvářena nabídka. O zákazníkovi budeme udržovat následující informace:

- identifikační číslo,
- jméno uživatele skládající se z titulu, jména a příjmení,
- telefon,
- fax,
- e-mail,
- datum narození.

Companies – zákazník může patřit do firmy, u které budeme požadovat uchování následujících informací:

- identifikační číslo,
- IČ,
- DIČ,
- telefon,
- fax,

- e-mail,
- číslo účtu.

Addresses – adresa skládající se z:

- město,
- ulice,
- poštovní směrovací číslo.

Bids – pro každého zákazníka může být vytvořeno několik nabídek a u každé nabídky budeme evidovat:

- identifikační číslo,
- datum, kdy byla uskutečněna,
- sleva na celou nabídku,
- platnost,
- jiná poznámka.

Templates – nabídka může být tvořena dříve vytvořenými šablonami. O šablonách budeme uchovávat pouze název.

Items – nabídka je tvořena několika položkami, o kterých budeme uchovávat:

- identifikační číslo,
- cena za kus,
- sleva na kus,
- množství,
- cena celkem.

Products – položka je tvořena fyzickými produkty. O každém produktu budeme chtít uchovávat informace:

- identifikační číslo,
- evidenční číslo,
- název,
- nákupní cena,
- prodejní cena,
- popis.

VAT – entita představující sazbu DPH. Uchovává pouze aktuální velikost DPH.

Subcategories – produkty jsou členěny do podkategorií. O každé podkategorii budeme uchovávat:

- identifikační číslo,
- název.

Categories – každá podkategorie patří do nadřazené kategorie. Stejně jakou u podkategorií budeme uchovávat:

- identifikační číslo,
- název.

4.2.2 ER-diagram systému

ER-diagram našeho systému je uveden na obrázku 4.1.

4.3 Use Case model

Use Case model definuje chování systému, aniž by odhaloval jeho vnitřní strukturu. Využití:

- specifikace požadavků na systém,
- komunikace se zákazníkem (uživatelem systému),
- podklad pro řízení projektu,
- tvorba testovacích případů pro fázi testování systému (před uvedením do provozu).

V našem informačním systému figurují tři role:

1. obchodní zástupce,
2. vedoucí výroby,
3. administrátor.

Obchodní zástupce

Obchodní zástupce představuje aktéra, který vytváří nabídky pro zákazníky.
Pravomoce obchodního zástupce jsou:

- Spravovat zákazníky
  Tato možnost zahrnuje vytváření nových zákazníků, spravovat stávající zákazníky nebo zákazníky odstraňovat.
- Spravovat firmy
  Každý uživatel může patřit do firmy. Obchodní zástupce může firmy přidávat, upravovat a mazat.
- Změnit osobní údaje
  Údaje o obchodním zástupci budou uvedeny v nabídce.
- Spravovat nabídky
  Obchodní zástupce může vytvářet nové nabídky, spravovat současné a mazat neaktuální. Nabídky je možné upravovat přidáním nebo smazáním položky.
- Odeslat nabídku e-mailem
  Vygeneruje nabídku do formátu pdf a odešle na zákazníkovu e-mailovou adresu.

Vedoucí výroby

Vedoucí výroby se stará o zadávání produktů do systému:

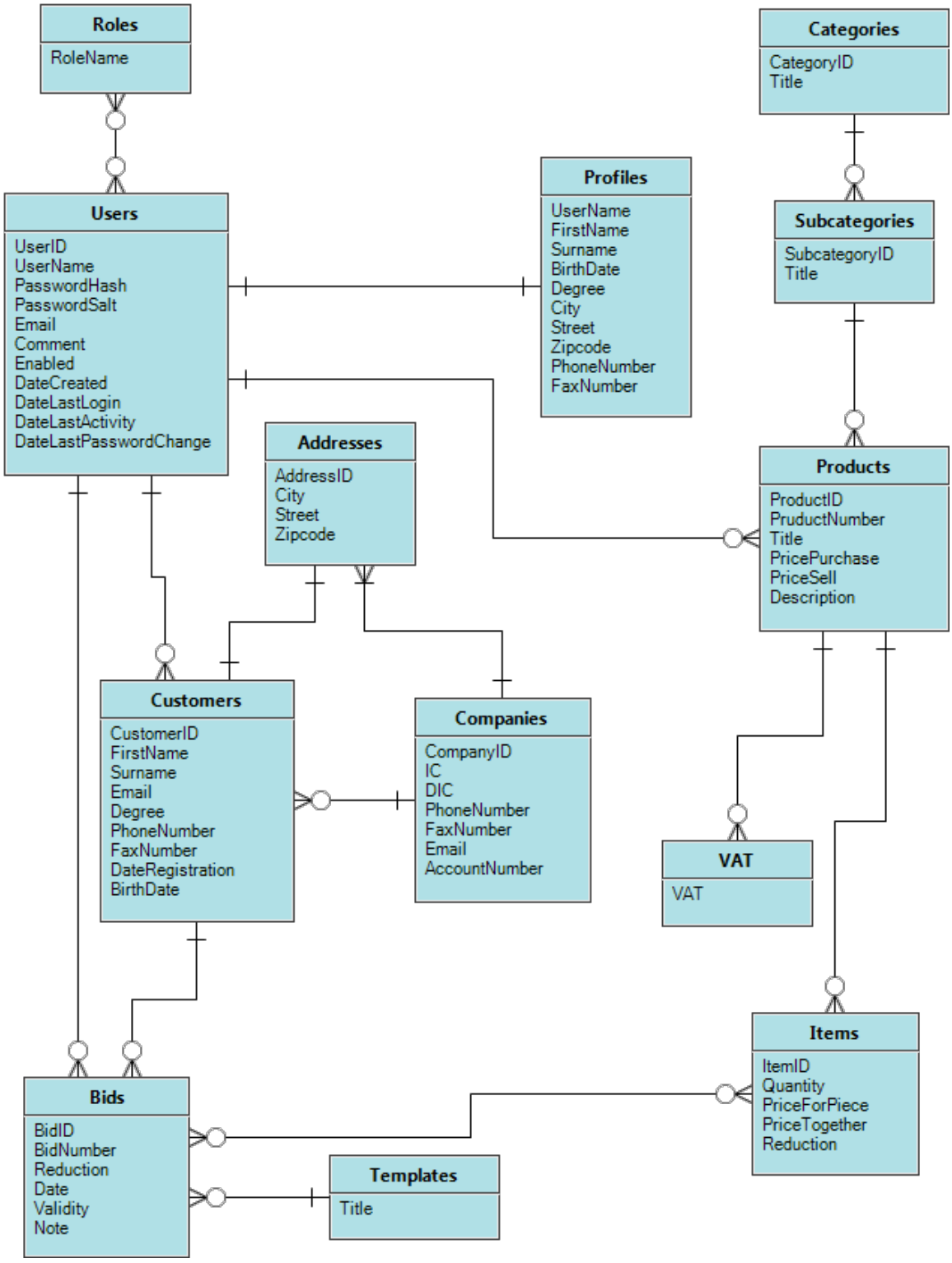
- Spravovat produkty
  Možnost přidávání, editování a mazání produktů a služeb.
- Spravovat kategorie
  Produkty lze řadit do kategorií a podkategorií.

Administrátor

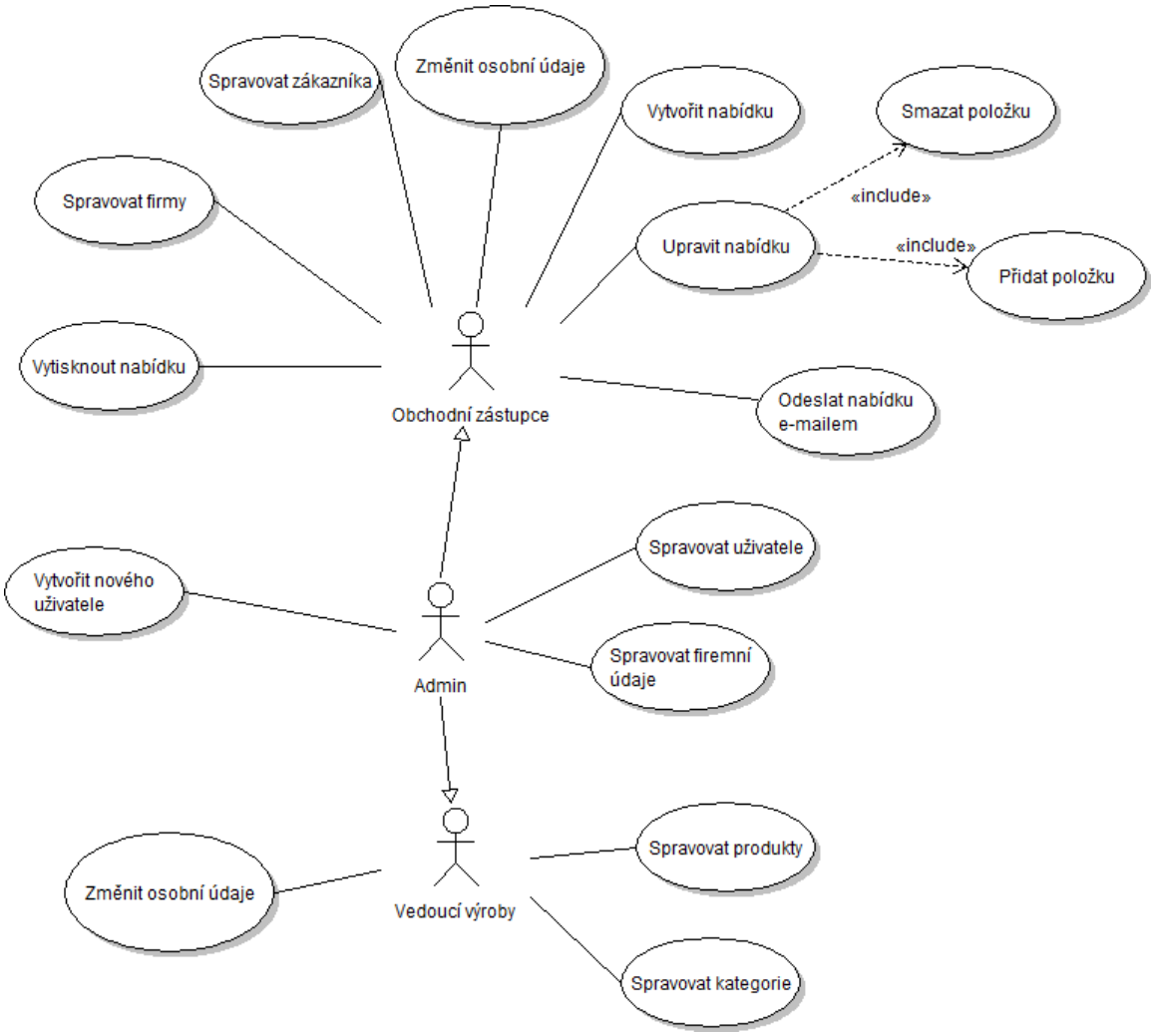
Administrátor má stejná práva jako obchodní zástupce a vedoucí výroby, navíc může:

- Vytvořit nového uživatele
  Přidá nového uživatele, následně jsou vyplněny uživatelské informace a role.
- Spravovat uživatele
  Lze měnit údaje stávajících uživatelů nebo mazat jejich účty.
- Spravovat firemní údaje
  Administrátor má práva měnit firemní údaje. Tyto údaje jsou zobrazeny v nabídce.

Use Case model systému je na obrázku 4.2.



Obrázek 4.1: Datábázový model systému



Obrázek 4.2: Use Case model

Kapitola 5

Implementace

Tato kapitola se zabývá popisem implementace webového systému. Podrobněji se zaměříme na databázovou část, strukturu aplikace, oprávnění a některé části systému.

5.1 Databáze

Při vytváření databáze se vycházelo z ER diagramu představeném v předchozí kapitole. Visual Studio 2008 obsahuje nástroje pro snadnou práci s SQL Serverem. Dalším možným způsobem vytvoření databázových tabulek je použít skript v příloze.

Pokud tabulka obsahuje primární klíč (často identifikační číslo), musí být prvek takto označen (PRIMARY KEY) je mu nastavena identita (IDENTITY), podle které se pak tomuto prvku přiřazuje automaticky hodnota při vkládání nového záznamu. Vazby mezi tabulkami jsou uskutečněny pomocí cizích klíčů (FOREIGN KEY). Pomocí klauzule ON DELETE CASCADE nebo ON UPDATE CASCADE lze zajistit kaskádové mazání záznamů, které obsahují hodnotu cizího klíče.

5.1.1 Připojovací řetězec

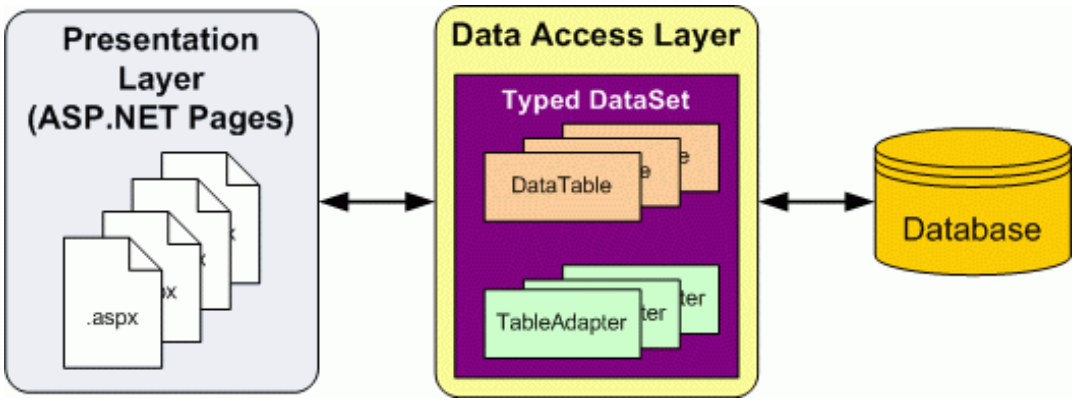
Protože v naší aplikaci používáme sadu providerů popsaných v sekci 4.1, každý vytvořený uživatel se ukládá do databáze. Aby toto aplikace umožňovala, musí být nastaven tzv. *připojovací řetězec*. Ten je nejčastěji uložen v konfiguračním souboru **web.config**. Výhodou uložení v tomto souboru je skutečnost, že v případě potřeby je pak změněn pouze na jednom místě. Nastavení připojení je znázorněno na obr. 5.1.

```
<appSettings>
  <add key="ConnectionStringName" value="BPConnectionString"/>
</appSettings>
<connectionStrings>
  <clear/>
  <add name="BPConnectionString" providerName="System.Data.SqlClient"
       connectionString="Data Source=B07-615B\sqlexpress;Initial Catalog=bt;Integrated Security=True"/>
  <add name="remote" providerName="System.Data.SqlClient"
       connectionString="server=192.168.1.5;uid=db2455;pwd=heslo;database=db2455"/>
</connectionStrings>
```

Obrázek 5.1: Nastavení připojovacího řetězce

5.2 Třívrstvá architektura

Pokud vytváříme osobní weby nebo nějaké menší webové aplikace, vystačíme si pouze s jednou vrstvou – **prezentační**. Taková vrstva potom zajišťuje jak připojení k databázi a získávání dat, tak i zobrazení některou webovou komponentou. V ostatních případech toto ovšem není vhodné, a proto musíme mít jednotlivé úkony rozděleny do více vrstev. Dvouvrstvá architektura tedy obsahuje navíc **datovou** vrstvu, která slouží k získávání dat z databáze. Ukázka vztahů mezi vrstvami je na obr. 5.2.



Obrázek 5.2: Ukázka vztahů ve dvouvrstvé architektuře (převzato z [8])

Toto členění odděluje datovou logiku od prezentační. Obvykle se nám ale hodí mít i nějakou logiku **aplikační**. Mnoho současných databázových aplikací má tedy celkem 3 vrstvy, které mezi sebou komunikují. Ukázka vztahů mezi těmito vrstvami je zobrazena na obr. 5.3.

Struktura

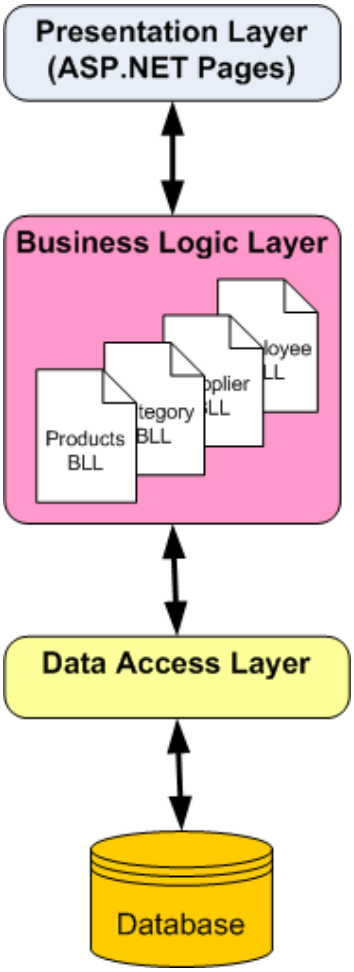
Naše aplikace je tvořena třemi vrstvami a její struktura je zobrazena na obr. 5.4.

5.2.1 Datová vrstva

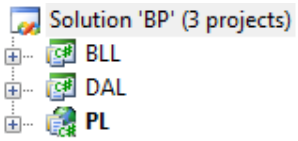
Datová vrstva (Data Access Layer – DAL) komunikuje s databází. Veškerý kód, který se týká datového zdroje (jako např. připojení k databázi, příkazy *SELECT*, *INSERT*, *UPDATE*, *DELETE*, atd.), by měl být umístěn v datové vrstvě. Toto oddělení je výhodné, pokud bychom např. chtěli později změnit datové úložiště nebo databázovou strukturu. Prezentační nebo aplikační vrstva tedy přímo s databází vůbec nekomunikuje.

DAL většinou obsahuje metody pro přístup k potřebným datům pomocí dotazů. Tyto metody jsou potom volány z aplikační vrstvy, připojí se k databázi, vykonají příslušný dotaz a vrátí výsledek.

V naší aplikaci je datová vrstva tvořena jako *Class Library* s názvem **DAL**. Do této knihovny je přidán *DataSet*. Ten představuje kolekci dat, která se skládá z instancí *DataTable* a každá tato instance obsahuje instanci *DataRow*. Pro každou databázovou tabulku (*Categories*, *Products*, *Customers*, ...) je vytvořen objekt *DataTable*. Protože *DataTable* neobsahuje informace o tom, jak přistupovat k databázi, využijeme třídu *TableAdapter*, která obsahuje metody volané z aplikační vrstvy.



Obrázek 5.3: Ukázka vztahů v třívrstvé architektuře (převzato z [7])

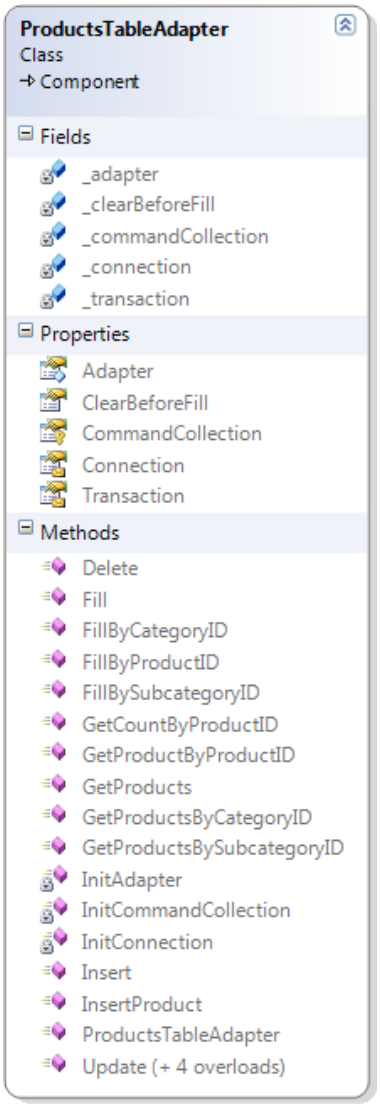


Obrázek 5.4: Struktura aplikace

Například databázovou tabulku *Products* tedy bude tvořit třída *ProductsTableAdapter* a bude mít metody jako `Delete`, `GetProducts`, `GetProductByProductID`, atd (obr. 5.5).

5.2.2 Aplikační vrstva

Aplikační vrstva (Business Logic Layer – BLL) tvoří prostředníka pro výměnu dat mezi datovou a prezentační vrstvou. Datová vrstva sice odděluje přístup k databázi od prezentační vrstvy, nedovoluje nám ale stanovit aplikační pravidla, která chceme v našem systému uplatnit. Je zde např. možné ošetřovat hodnoty z formulářů zadávané do databáze. Tuto



Obrázek 5.5: Třída *ProductsTableAdapter*

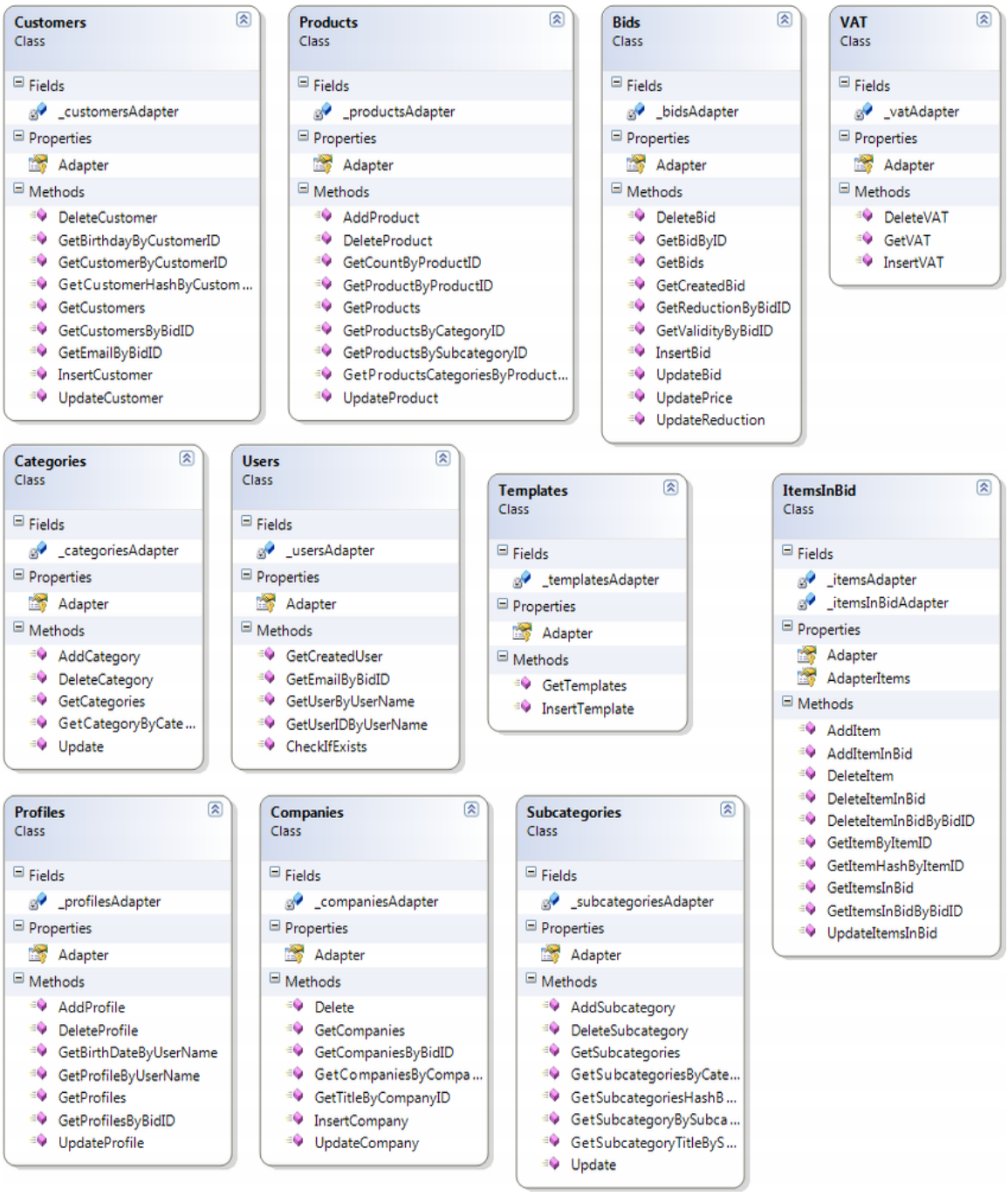
činnost ale v našem systému budeme provádět v prezenční vrstvě pomocí tzv. *Validatorů*[1]. Dalším příkladem uplatnění aplikačního pravidla může být situace, kdy nechceme z databáze odstranit zákazníka, pro kterého je v systému vytvořena alespoň jedna nabídka.

V naší aplikaci představuje aplikační vrstvu *Class Library* s názvem **BLL**. Pro každý *TableAdapter* vytvořený v datové vrstvě je zde vytvořena třída, která bude obsahovat metody pro pro vkládání, mazání, aktualizování tabulek stejně jako *TableAdapter* s tím, že budou uplatněna naše aplikační pravidla. Tyto metody jsou potom volány z prezentační vrstvy.

[1] V naší aplikaci budeme využívat *RequiredFieldValidator*, *CompareValidator* a *RegularExpressionValidator*.

Diagram tříd

Obr. 5.6 znázorňuje diagram tříd v aplikační vrstvě.



Obrázek 5.6: Diagram tříd v aplikační vrstvě

5.2.3 Prezentační vrstva

Většina dnešních aplikací vyžaduje nějakou úroveň uživatelské interakce. Vrstva, která zajišťuje interakci, výsledný vzhled, přijímání a validaci vstupních údajů aplikace nebo také získávání dat z aplikační či datové vrstvy, se nazývá prezentační vrstva (Presentation Layer – PL). Prezentační vrstva může být tvořena např. pomocí ASP.NET *Web Forms* nebo technologií *Windows Forms*.

V našem systému představuje prezentační vrstvu *ASP.NET Web Application* s názvem **PL**.

5.2.4 Získávání dat

Pokud tedy chceme získat data v prezentační vrstvě, musíme vytvořit objekt některé třídy v aplikační vrstvě a zavolat jeho metodu. Tento objekt vytvoří objekt příslušné třídy v datové vrstvě a zavolá metodu objektu v této vrstvě.

Prezentační vrstva

Chceme-li například získat všechny produkty v naší databázi a zobrazit je třeba v ovládacím prvku *GridView*, máme v prezentační vrstvě 2 možnosti:

- Napojit data na *GridView* v *Code Behind* nebo
- přidat na stránku komponentu *ObjectDataSource*.

V prvním případě musíme přidat referenci na třídy v knihovně *BLL* (obsahuje třídy aplikační vrstvy) použitím `using BLL;`. To nám umožní vytvořit objekt třídy *Products* a zavolat metodu `GetProducts`, která vrátí všechny produkty jako objekt třídy *ProductsDataTable*. Tento objekt pak zvolíme jako zdroj dat pro *GridView* (obr. 5.7).

```
Products products = new Products();
GridView1.DataSource = products.GetProducts();
GridView1.DataBind();
```

Obrázek 5.7: Získání dat v code behind

Další možností je přidat na stránku prvek *ObjectDataSource* (obr. 5.8). Tomuto prvku je pak nutné nastavit hodnotě atributu *SelectMethod* název metody, kterou chceme použít (v tomto případě `GetProducts`). Dále je potřeba atributu *TypeName* přiřadit třídu, jejíž metodu chceme použít (`BLL.Products`). Posledním krokem je přidat atributu *DataSourceID* v komponentě *GridView* stejnou hodnotu, jako má atribut *ID* prvku *ObjectDataSource*.

```
<asp:ObjectDataSource ID="ObjectDataSource2" runat="server"
    OldValuesParameterFormatString="{0}" SelectMethod="GetProducts"
    TypeName="BLL.Products">
</asp:ObjectDataSource>
```

Obrázek 5.8: Získání dat pomocí *ObjectDataSource*

Aplikační vrstva

Metoda `GetProducts` pouze volá metodu stejného názvu třídy *ProductsTableAdapter* a vrací objekt třídy *ProductsDataTable*. Navíc je zde uveden před funkcí parametr `DataObjectMethodAttribute`, který nám usnadní výběr metod, které chceme použít, pokud používáme průvodce pro nastavení konfigurace komponenty *ObjectDataSource*.

```
[System.ComponentModel.DataObjectMethodAttribute(
    System.ComponentModel.DataObjectMethodType.Select, true)]
public BiddingSystemDB.ProductsDataTable GetProducts()
{
    return Adapter.GetProducts();
}
```

Obrázek 5.9: Získání dat v aplikační vrstvě

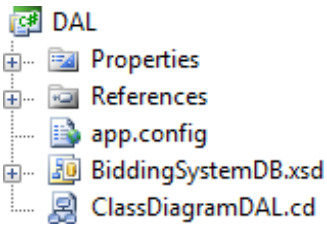
Datová vrstva

Metoda `GetProducts` v datové vrstvě zavolá příslušný dotaz na databázi a vrátí výsledek do vrstvy aplikační. V tomto případě má dotaz podobu:
`SELECT ProductID, ProductNumber, Title, PricePurchase, PriceSell, Description, CategoryID, SubcategoryID, UserID, (SELECT Title FROM Categories WHERE Categories.CategoryID = Products.CategoryID) AS Category, (SELECT Title FROM Subcategories WHERE Subcategories.SubcategoryID = Products.SubcategoryID) AS Subcategory, (SELECT UserName FROM Users WHERE Users.UserID = Products.UserID) AS UserName FROM Products`[2].

5.3 Struktura aplikace

Datová vrstva

Struktura datové vrstvy je zobrazena na obr. 5.10.



Obrázek 5.10: Struktura datové vrstvy

Properties Obsahuje soubor AssemblyInfo.cs, který obsahuje informace o aplikaci, jako např. číslo verze, popis, atd.

[2] V tomto příkazu by pro získání dat z více tabulek bylo vhodnější použití klauzule *JOIN*. Pokud ale každý údaj z cizí tabulky získáme *SELECT* dotazem, Visual Studio nám umožní automaticky vygenerovat příkazy pro přidání prvku do tabulky nebo smazání či aktualizování prvku, což se nám v tomto případě hodí.

References Obsahuje reference na použité třídy.

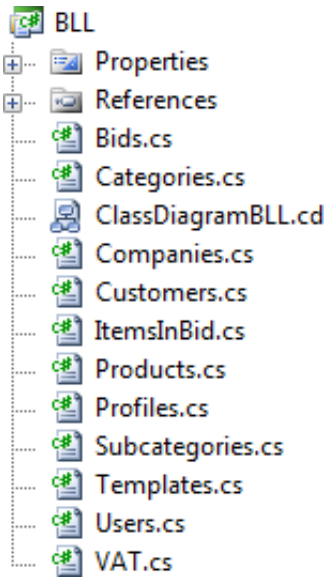
app.config Zde je uložen *ConnectionString* pro všechny *Table Adapter*y, aby nemusel být u každého uložen zvlášť. V případě potřeby je pak změněn pouze v tomto souboru.

BiddingSystemDB.xsd *DataSet*, který se skládá z instancí *DataTable* vytvořenou pro každou databázovou tabulku.

ClassDiagramDAL.cd Digram tříd reprezentující datovou vrstvu.

Aplikační vrstva

Struktura datové vrstvy je zobrazena na obr. 5.11.



Obrázek 5.11: Struktura aplikační vrstvy

Datová vrstva obsahuje třídy reprezentující databázové tabulky. Dále obsahuje diagram tříd reprezentující tuto vrstvu.

Prezentační vrstva

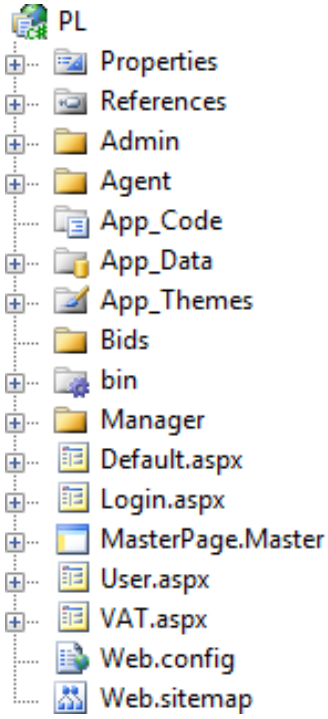
Struktura prezentační vrstvy je zobrazena na obr. 5.12.

Properties Obsahuje soubor AssemblyInfo.cs, který obsahuje informace o aplikaci, jako např. číslo verze, popis, atd.

References Obsahuje reference na použité třídy.

Admin Obsahuje soubory, ke kterým má přístup pouze administrátor.

- Company.aspx – zobrazuje informace o firmě. Je zde možné tyto informace měnit.
- Company.xml – konfigurační soubor obsahující informace o firmě.
- CreateUser.aspx – umožňuje vytvořit nového uživatele.



Obrázek 5.12: Struktura aplikace

- Default.aspx – defaultní stránka administrace. Obsahuje odkazy na administrační stránky.
- UserDetail.aspx – zobrazuje informace o uživateli. Je možné zde měnit jeho údaje.
- Users.aspx – zobrazuje seznam uživatelů.

Agent Obsahuje soubory, ke kterým má přístup obchodní zástupce a administrátor.

- Bid.aspx – zobrazuje informace o nabídce. Dále umožňuje upravovat nabídku nebo odeslat nabídku e-mailem.
- BidDetail.aspx – umožňuje vytvořit novou nabídku nebo upravit stávající.
- bidinfo.xslt – šablona pro tvorbu pdf souboru.
- Bids.aspx – zobrazuje seznam nabídek.
- Companies.aspx – zobrazuje seznam firem.
- CompanyDetail.aspx – umožňuje vytvářet a upravovat firmy.
- CreateTemplate.aspx – umožňuje vytvořit šablonu z nabídky.
- CustomerDetail.aspx – umožňuje vytvářet a upravovat zákazníky.
- Customers.aspx – zobrazuje seznam zákazníků.
- LoadTemplate.aspx – umožňuje načíst uloženou šablonu nabídky.
- SendBid.aspx – umožňuje odeslat nabídku e-mailem.

App_Code Obsahuje soubory, které definují třídy použitelné v celé aplikaci.

App_Data Obsahuje uložené soubory.

App_Themes Obsahuje soubory, které definují vzhled aplikace.

- Default
  - Images – obsahuje obrázky použité v aplikaci.
  - Skin.skin – obsahuje skiny pro ovládací prvky v aplikaci.
  - Stylesheet.css – obsahuje kaskádové styly pro vzhled aplikace.

Bids Do této složky jsou ukládány vytvořené nabídky v podobě pdf.

Bin Obsahuje zkompilované .dll soubory potřebné pro běh aplikace.

Manager Obsahuje soubory, ke kterým má přístup vedoucí výroby a administrátor.

Default.aspx Defaultní stránka aplikace.

Login.aspx Umožňuje přihlášení do aplikace.

MasterPage.master Definuje rozložení a vzhled stránky.

User.aspx Umožňuje zobrazit a měnit uživatelské údaje.

VAT.aspx Umožňuje zobrazit a upravit velikost DPH.

Web.config Konfigurační soubor aplikace.

Web.sitemap Představuje navigační menu aplikace.

5.4 Oprávnění

Visual Studio 2008 obsahuje webovou aplikaci, která umožňuje vytváření uživatelů, rolí a nastavení oprávnění. Tato aplikace se jmenuje ASP.NET Configuration Tool (Nástroj pro správu webu) a je dostupná po kliknutí na ikonu Solution Explorer. V sekci zabezpečení je tedy možné vytvořit uživatele. Vytvoříme 3 pokusné uživatele[3] (admin, agent, manager) a nastavíme jim role a práva přístupu podle struktury v sekci 5.3.

Protože užíváme providery popsané v sekci 4.1, budou tito uživatelé a role vloženy do příslušných databázových tabulek. Lze tedy jednoduše přidávat uživatele i bez této webové aplikace pomocí SQL skriptů.

Nastavení oprávnění se muselo taktéž někam promítnout, a to do konfiguračního souboru **web.config** do části *system.web*. Soubor **web.config** se taktéž vytvořil ve složkách uživatelů (Admin, Agent, Manager) a obsahuje oprávnění pro tyto uživatele.

```
<authorization>
    <deny users="?" />
</authorization>
```

Obrázek 5.13: Nastavení oprávnění v souboru web.config

V souboru **web.config** dále upravíme autorizaci pomocí *forms*, tedy jménem a heslem. Standardně se ověřuje pomocí přihlášení do Windows, což se nám může hodit, pokud by aplikace „běžela“ na Intranetu.

[3] Další možností je vytvořit pouze administrátora, protože ten má možnost v našem systému vytvářet nové uživatele.

5.5 Vzhled aplikace

MasterPage je speciální stránka (v našem případě **MasterPage.master**) obsahující layout a dynamickou část, která je reprezentována komponentou *ContentPlaceHolder*. Do této komponenty je dynamicky dosazován obsah souborů, na které v url odkazujeme. Celý dynamický obsah musí být uvnitř komponenty *Content*, které navíc musíme nastavit hodnotu vlastnosti *ContentPlaceHolderId* na ID *ContentPlaceHolder*u v **MasterPage**, do kterého dosazujeme. Navíc v direktivě *Page* této stránky musíme specifikovat vlastnost *MasterPageFile*, která ukazuje na příslušný **MasterPage**.

```
<div id="content">
    <div id="left">
        <asp:Label ID="bulletedList" runat="server"></asp:Label>
        <asp:SiteMapDataSource ID="siteMapData" runat="server" ShowStartingNode="False" />
    </div>
    <div id="right">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    <div style="clear: both;">
    </div>
</div>
```

Obrázek 5.14: Ukázka vytvoření MasterPage s komponentou ContentPlaceHolder

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.Master" AutoEventWireup="true"
    CodeBehind="Default.aspx.cs" Inherits="BP.WebForm16" Title="Vítejte" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    Vítejte
</asp:Content>
```

Obrázek 5.15: Začlenění obsahu do MasterPage

5.5.1 Menu

Položky menu jsou obsaženy v souboru **Web.sitemap**. Jedná se o XML soubor obsahující značky *siteMapNode*. Každá značka představuje jednu položku, přičemž značky do sebe mohou být libovolně zanořovány (obr. 5.16).

```
<siteMapNode url="~/Admin/Default.aspx" title="Administrace" description="" roles="Admin">
  <siteMapNode url="~/Admin/Company.aspx" title="Firma" description="" />
  <siteMapNode url="~/Admin/CreateUser.aspx" title="Vytvořit uživatele" description="" />
  <siteMapNode url="~/Admin/Users.aspx" title="Uživatelé" description="" />
</siteMapNode>
```

Obrázek 5.16: Konfigurace menu

Tento soubor je pak zpracován komponentou *SiteMapDataSource*, která ho zobrazí jako seznam. Výhodou této implementace je, že přihlášenému uživateli se zobrazí menu automaticky podle nastavených práv a nemusíme se o to starat ručně. Každý uživatel bude mít tedy

menu sestaveno z jiných položek. Kompletní menu pro uživatele v roli **administrátora** je na obr. 5.17.

Obrázek 5.17: Ukázka menu

5.6 Správa produktů

Jednou z významných částí systému je evidence firemních produktů a služeb. O zadávání nových nových produktů, aktualizování stávajících nebo odstraňování produktů se stará uživatel v roli vedoucího výroby.

Seznam produktů je zobrazen na stránce **Products.aspx** komponentou *GridView* (obr. 5.18). Na této stránce je možné produkty odstraňovat. Po stisknutí tlačítek je uživatel přesměrován na stránku **ProductDetail.aspx**, kde může přidat do systému nový produkt nebo upravit produkt stávající.

Při úpravě nebo vkládání produktu je zajištěno, aby byl zadán správný formát vstupních údajů. U položek, kde je potřeba, aby byly vyplněny, jsou použity komponenty *RequiredFieldValidator*, které nepovolí odeslat formulář, pokud není údaj zadán. U položek *Nákupní cena* a *Prodejní cena* je navíc ovládací prvek *CompareValidator*, který porovnává zadanou hodnotu nebo typ zadané hodnoty s údaji zadanými v tomto prvku. U ceny požadujeme, aby byla zadaná hodnota typu *double*.

Číslo produktu	Název	Nákupní cena	Prodejní cena	Popis	Vložil	Kategorie		
1/1	Transsteel 3500		35 000,00 Kč	robustní a spolehlivý velmi snadné ovládání perfektní svařování oceli	admin	Obloukové svařování Svařovací zdroje		
1/2	VarioStar 1500		30 000,00 Kč	robustní konstrukce stupňovité spínání do 150 A pro svařování tenkých plechů	admin	Obloukové svařování Svařovací zdroje		
2/1	AL2300		50 000,00 Kč	vzduchem chlazený použití v ocelářském průmyslu	admin	Obloukové svařování Svařovací hořáky		

Obrázek 5.18: Přehled produktů

5.7 Správa zákazníků

Pokud chce vytvořit obchodní zástupce nabídku, musí nejprve do systému vložit zákazníka, pro kterého bude nabídka tvořena.

Přehled zákazníků je zobrazen na stránce **Customers.aspx** komponentou *GridView*. Protože v našem systému uchováváme o zákaznících poměrně hodně informací, nejsou na této stránce uvedeny všechny. Přehled všech údajů je možné zobrazit po kliknutí na informační tlačítko – dojde k přesměrování na stránku **CustomerDetails.aspx**. Na tomto místě je možné také upravit zákazníkovy údaje nebo vytvořit nového zákazníka.

K zobrazení i úpravám údajů slouží komponenta *FormView*, která je „napojena" na komponentu *ObjectDataSource*. Pohodlnější změna data narození je umožněna přidáním dvou komponent *DropDownList* pro výběr měsíce a roku, den narození je pak vybrán v komponentě *Calendar*. Vyhneme se tím také případu, kdy by uživatel zadal nesprávný formát a ten kolidoval s databází.

Všechny položky jsou taktéž zabezpečeny proti zadání nesprávného formátu nebo nevyplnění pole v případě, kdy to potřebujeme.

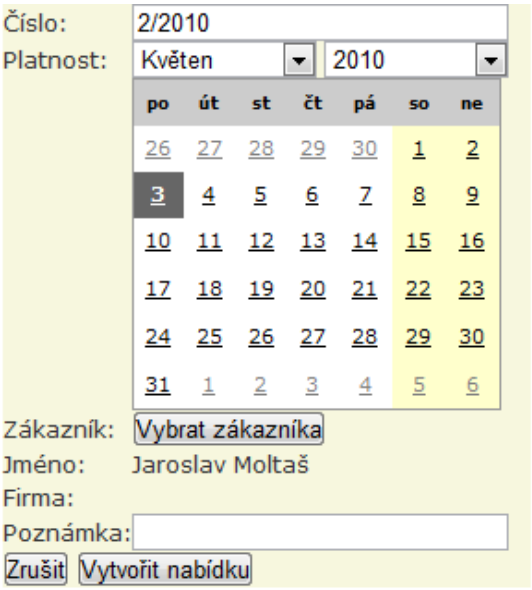
- **Jméno** – je povinné, a proto je na prvek *TextBox* „nasměrován" prvek *RequiredFieldValidator*, který se stará o to, aby formulář nešel odeslat, pokud je toto pole prázdné.
- **Příjmení** – je taktéž povinné a ošetřeno proti nevyplnění tohoto pole.
- **Datum narození** – povinné. Vzhledem k tomu, že toto datum bude převzato z prvku *Calendar* (kde je implicitně nějaké datum označeno), nemusíme se starat o to, že by uživatel datum nezadal nebo zadal nesprávný formát.
- **E-mail** – povinný. Tento *TextBox* je ošetřen kromě prvku *RequiredFieldValidator* také prvkem *RegularExpressionValidator*, který porovná zadanou hodnotu s regulárním výrazem, který má tento prvek nastavený. V tomto případě jsme použili výraz `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$`. Tento regulární výraz povolí pouze platný formát e-mailu. Není však ověřena jeho existence.
- **Telefon** – povinné. V tomto případě je také použit prvek *RegularExpressionValidator*, abychom zajistili vložení správného formátu čísla. Výraz `^(\+420)? ?[0-9]{3} ?[0-9]{3} ?[0-9]{3}$` povolí uživateli zadat číslo s mezinárodní předvolbou i bez ní, poté následují 3 skupiny po třech číslech, mezi nimiž může (ale nemusí) být mezera.
- **Fax** – nepovinný. Pokud je ale zadán, formát čísla odpovídá stejným kritériím jako telefonní číslo.
- **Město** – povinné a ošetřené, aby bylo zadáno.
- **Ulice a č.p.** – povinná.
- **PSČ** – povinné. Ošetřeno prvkem *RegularExpressionValidator* s regulárním výrazem `^\d{3} ?\d{2}$`, který bude odpovídat jak číslu zadanému ve formátu 12345, tak číslu 123 45[4].

[4] Místo prvku *RegularExpressionValidator* bychom zde také mohli použít prvek *RangeValidator*, který kontroluje, zda zadaná hodnota patří do rozsahu, který má tento prvek nastavený. Rozsah by byl nastaven např. na 10000-99999. Použití *RegularExpressionValidator* je však v tomto případě vhodnější a pružnější.

5.8 Tvorba nabídek

Vytváření nabídek je nejdůležitější částí systému. Přes odkaz v menu *Vytvořit nabídku* se uživatel dostane na stránku **BidDetail.aspx**, která slouží právě k tomuto účelu. Zde uživatel vyplní tyto položky:

- **Číslo** – číslo nabídky je povinné.
- **Platnost** – datum, dokdy nabídka platí. Výběr je realizován pomocí dvou *DropDownListů* a prvku *Calendar*.
- **Zákazník** – po kliknutí na tlačítko *Vybrat zákazníka* se pod formulářem s nabídkou zobrazí seznam zákazníků. Pomocí potvrzovacího tlačítka uživatel vybere zákazníka a tím se jeho údaje zobrazí i ve formuláři s nabídkou.
- **Poznámka** – zde uživatel může vyplnit nějakou poznámku či komentář k nabídce.

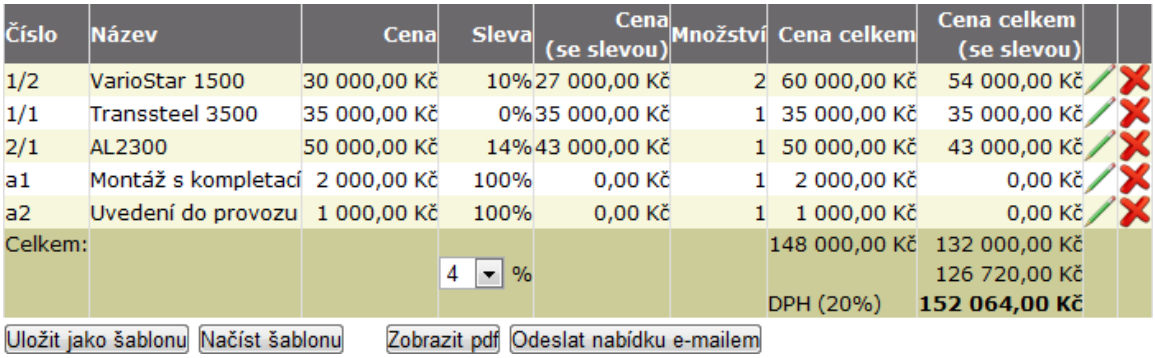


Titul	Jméno	Příjmení	E-mail	Město	Ulice	Tel. číslo	Firma	
	Jaroslav	Moltaš	jar.zima@seznam.cz	Oponešice	Oponešice 26	728952537		✓
	Jan	Novák	jan.novak@abc.com	Praha	Hlavní 298	+420 123 456 789		✓
ing.	Petr	Dočkal	petr@dockal.cz	Brno	Soukenická 56	+420 987 654 321		✓
	Antonín	Novotný	antonin@novotny.com	Pelhřimov	Hradní 12	234567890	Kotrba	✓

Obrázek 5.19: Tvorba nabídky

Po vytvoření nabídky se uživatel dostává na stránku **Bid.aspx**. V horní části jsou zobrazeny informace o nabídce . Tyto informace je možné upravit přechodem na stránku **BidDetail.aspx** po stisknutí tlačítka *Upravit*.

Pod těmito informacemi se nachází komponenta *GridView*, která zobrazuje položky v nabídce (obr. 5.20). Je zde možné tyto položky upravovat, mazat nebo přidávat nové. V zápatí tohoto seznamu položek je také možnost zvolit slevu na celou nabídku.



Číslo	Název	Cena	Sleva	Cena (se slevou)	Množství	Cena celkem	Cena celkem (se slevou)		
1/2	VarioStar 1500	30 000,00 Kč	10%	27 000,00 Kč	2	60 000,00 Kč	54 000,00 Kč		
1/1	Transsteel 3500	35 000,00 Kč	0%	35 000,00 Kč	1	35 000,00 Kč	35 000,00 Kč		
2/1	AL2300	50 000,00 Kč	14%	43 000,00 Kč	1	50 000,00 Kč	43 000,00 Kč		
a1	Montáž s kompletací	2 000,00 Kč	100%	0,00 Kč	1	2 000,00 Kč	0,00 Kč		
a2	Uvedení do provozu	1 000,00 Kč	100%	0,00 Kč	1	1 000,00 Kč	0,00 Kč		
Celkem:			4 %			148 000,00 Kč DPH (20%)	132 000,00 Kč 126 720,00 Kč **152 064,00 Kč**		

Uložit jako šablonu | Načíst šablonu | Zobrazit pdf | Odeslat nabídku e-mailem

Obrázek 5.20: Položky v nabídce

5.8.1 Přidávání a editace položek

Uživateli se zobrazí seznam produktů, ze kterého si může vybrat ten, který chce přidat do nabídky. Poté, co je produkt vybrán, uživatel zadá množství a slevu. Velikost slevy je určena tzv. slevovým prostorem[5].

5.9 Generování PDF

V této sekci se zaměříme na to, jaké nástroje a součásti potřebuje pro převod nabídky do formátu pdf. Generování probíhá v případě, že uživatel chce vidět náhled nabídky (tlačítko *Zobrazit pdf*) nebo chce nabídku odeslat e-mailem (tlačítko *Odeslat nabídku e-mailem*).

5.9.1 NFOP

NFop je formátovací procesor (Formatting Objects Processor – FOP) pro XSL-FO, který „běží“ na platformě .NET Framework. Tvoří jakýsi přechod z Apache XML projektu FOP, určeného pro Javu, k .NET Visual J# [2].

Proto je nutné mít nainstalovaný balíček *Microsoft J#.NET redistributable package*. Ve Visual Studiu je navíc potřeba přidat reference na tyto knihovny.

5.9.2 XML

Další věcí, kterou budeme potřebovat, jsou data, která budeme chtít zobrazovat, ve formátu XML. V okamžiku, kdy uživatel požaduje vygenerování pdf, je v paměti vytvořen XML dokument, do kterého jsou načtena všechna potřebná data (obr. 5.21). Těmi jsou údaje o:

- **Nabídce** – číslo, datum, platnost, celková cena, sleva na nabídku, DPH,...
- **Naší firmě** – název, adresa, e-mail, telefonní číslo,...
- **Zákazníkovi** – jméno, adresa, e-mail, telefonní číslo,...
- **Firmě zákazníka** – název, adresa, e-mail, telefonní číslo,...

[5] Slevový prostor je dán velikostí nákupní a prodejní ceny. Pokud nákupní cena chybí, je slevový prostor v rozmezí 0-100%. Pokud je nákupní cena uvedena, je slevový prostor v rozmezí 0-max, kde max představuje hodnotu vypočítanou ze vzorce $\frac{prodejni\_cena - nakupni\_cena}{prodejni\_cena} * 100$.

- **Obchodním zástupci** – jméno, adresa, e-mail, telefonní číslo,...
- **Položkách nabídky** – číslo, název, cena, sleva,...

```
<?xml version="1.0" encoding="utf-8"?>
<Bid>
  <Info>
    <Number>2/2010</Number>
    <Validity>3. května 2010</Validity>
    ...
  </Info>
  <Company>
    <Title>Komandus</Title>
    <Email>komandus@komandus.cz</Email>
    ...
  </Company>
  <Customer>
    <Name>Jaroslav Moltaš</Name>
    <Email>jar.zima@seznam.cz</Email>
    ...
  </Customer>
```

Obrázek 5.21: Ukázka dat

5.9.3 XSL

Poslední částí potřebnou k transformaci je xsl šablona (soubor **bidinfo.xsl**), která určí výslednou strukturu a vzhled souboru pdf.

Tato šablona obsahuje tag `<xsl:template match="/Bid"/>`. Hodnota atributu *match* představuje kořenový element XML souboru. Jakmile procesor XSL narazí na tag `<xsl:value-of select=""/>`, dosadí na toto místo hodnotu prvku ze souboru XML, který odpovídá atributu *select*. Pokud tedy budeme mít např. tag `<xsl:value-of select="Company/Title"/>`, bude dosazená hodnota „převzata“ z prvku, který má hierarchii `<Bid><Company><Title>hodnota</Title></Company></Bid>`.

Rozměry a okraje

Rozměry a okraje stránky jsou definované v tzv. *Page Master*u, který představuje tag `<fo:simple-page-master />`. Těchto *Page Master*ů může být v jedné šabloně více a každý musí mít své unikátní jméno (atribut *master-name*). Jsou zde nastavené rozměry stránky, okraje a velikost záhlaví (`<fo:region-before />`), těla (`<fo:region-body />`) a zápatí (`<fo:region-after />`). Všechny tyto předpisy najdeme uvnitř `<fo:layout-master-set>`.

Obsah

Začátek obsahu je uvozen tagem `<fo:page-sequence />`. Jeho hodnota atributu *master-reference* musí odpovídat hodnotě atributu *master-name*, kterou jsme uvedli v tagu `<fo:simple-page-master />`.

XSL šablony umožňují vytvářet rozvržení ve stylu záhlaví–tělo–zápatí. V **záhlaví** (obr. 5.22) každé stránky budeme chtít zobrazit informace o naší firmě. K tomu použijeme předpis

`<fo:static-content />` s hodnotou atributu *flow-name* `xsl-region-before`. Jednotlivé hodnoty jsou uvnitř bloku `<fo:block />`.

```
<!-- firma -->
<fo:static-content flow-name="xsl-region-before">
  <fo:block font-family="Arial"
            font-weight="bold"
            font-size="15pt">
    <xsl:value-of select="Company/Title"/>
  </fo:block>
  <fo:block font-family="Arial">
    <xsl:value-of select="Company/City"/>
    <xsl:value-of select="Company/Street"/>
    <xsl:value-of select="Company/Zipcode"/>
  </fo:block>
  <fo:block font-family="Arial">
    <xsl:value-of select="Company/Email"/>
  </fo:block>
  <fo:block font-family="Arial">
    <xsl:value-of select="Company/PhoneNumber"/>
  </fo:block>
</fo:static-content>
```

Obrázek 5.22: Ukázka xsl předpisu pro zobrazení informací o firmě

Zápatí je opět uvozeno tagem `<fo:static-content />` s hodnotou atributu *flow-name* `xsl-region-after` a bude obsahovat IČO a DIČ naší firmy.

Tělo stránky obsahuje informace o:

- Nabídce,
- zákazníkovi,
- obchodním zástupci,
- firmě a
- položkách nabídky.

Ukázka nabídky v pdf je na obr. 5.23.

5.10 Odeslání e-mailem

Pokud chce uživatel odeslat nabídku e-mailem, soubor pdf se uloží do složky **Bids**. Poté je uživatel přesměrován na stránku **SendBid.aspx**, kde je komponenta *TextBox* pro vyplnění e-mailové adresy příjemce. Toto pole je standardně předvyplněno zákazníkovou e-mailovou adresou a je chráněno proti vyplnění neplatným formátem e-mailu komponentou *RequiredFieldValidator*. Po stisknutí tlačítka *Odeslat* je nabídka odeslána jako příloha na uvedenou adresu[6].

[6] Pro správné odeslání je potřeba korektní nastavení SMTP serveru v souboru **SendBid.aspx.cs** (položky `host` a `port`).

Komandus
Praha, Králova 7, 118 00
komadus@komandus.cz
+420 123 456 789

Nabídka c. 2/2010
Datum: 8. kvetna 2010
Platnost: 21. kvetna 2010
Poznámka:

Zákazník
Jméno: Jan Novák
Adresa: Praha, Hlavní 298, 111 22
E-mail: jan.novak@abc.com
Telefon: +420 123 456 789

Nabídku vytvoril
Jméno: Jaroslav Moltaš
Adresa: Oponešice, Oponešice 26, 67532
E-mail: admin@admin.cz
Telefon: 728952537

Firma
Název: Kotrba
Adresa: Mladá Boleslav, Sokolská 90, 999 88
E-mail: kotrba@kotrba.cz
Telefon: +420 123 456 656

Číslo / název		Cena / ks	Množství	Cena celkem	Netto
1/1		35 000,00	1	35 000,00	
Transsteel 3500	- 10%	31 500,00		31 500,00	31 500,00
2/1		50 000,00	1	50 000,00	
AL2300	- 10%	45 000,00		45 000,00	45 000,00
40/123		700,00	2	1 400,00	
Bunda	- 0%	700,00		1 400,00	1 400,00
40/321		400,00	1	400,00	
Vesta	- 0%	400,00		400,00	400,00
25/2		200,00	10	2 000,00	
Rukavice High End	- 0%	200,00		2 000,00	2 000,00
a1		2 000,00	1	2 000,00	
Montáž s kompletací	- 100%	0,00		0,00	0,00
a2		1 000,00	1	1 000,00	
Uvedení do provozu	- 100%	0,00		0,00	0,00
Celkem				**CZK**	**80 300,00**
				- 4%	**77 088,00**
			DPH	**20%**	15 417,60
Celkem s DPH				**CZK**	**92 505,60**

Obrázek 5.23: Ukázka nabídky v PDF

Kapitola 6

Závěr

V této práci jsme si popsali rozbor a popis tvorby webové aplikace v ASP.NET. Aplikace slouží jako informační systém firmy, která eviduje své produkty a služby a z nich vytváří nabídky pro zákazníky. Tyto nabídky je následně možné vytisknout či odeslat zákazníkovi e-mailem.

Jedním z možných rozšíření aplikace by mohlo být provázání produktů. Funkčnost by byla přibližně následující: obchodní zástupce by vybral produkt a s ním by přibyl do nabídky produkt jiný, např. se 100% slevou. V případě naší firmy, která je zaměřena na svařovací techniku, by bylo uplatnění např. takové, že ke každé svářečce by byla montáž a zaškolení zdarma. V současném stavu aplikace je toto možné, ovšem pouze manuálně.

Dalším možným rozšířením by bylo zobrazení cen v jiné měně, než v českých korunách. Navíc by kurzy mohly být aktualizovány automaticky z nějaké internetové databáze.

Díky této práci jsem se blíže seznámil s technologií ASP.NET a jejím třívrstvým modelem a zjistil tak, jaké pohodlí přináší při programování www stránek např. oproti technologii PHP. Také jazyk C# mi přišel velmi komfortní. Kvůli četným problémům, které jsem musel v průběhu práce řešit, jsem pročetl mnoho publikací a diskuzních fór o těchto technologiích. Díky tomu jsem si prohloubil znalosti a rád bych se vývoji aplikací v ASP.NET věnoval i v budoucnu.

Literatura

[1] Cay S. Horstmann: Violet UML Editor [online]. http://alexdp.free.fr/violetumleditor/page.php, 2010 [cit. 2010-01-11].

[2] Geeknet, Inc.: NFop [online]. http://sourceforge.net/projects/nfop/, 2010 [cit. 2010-04-22].

[3] Marek Běhálek: Architektura .NET Framework [online]. http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch01s01.html, 2007 [cit. 2010-04-22].

[4] Marek Běhálek: XSLT [online]. http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch08s06.html, 2007 [cit. 2010-04-22].

[5] Microsoft: Altairis Web Security Toolkit [online]. http://altairiswebsecurity.codeplex.com/, 2006-2010 [cit. 2010-01-11].

[6] Microsoft Corporation: Microsoft Visual Studio: přehled [online]. http://www.microsoft.com/cze/msdn/produkty/vstudio/default.mspx, 2010 [cit. 2010-01-11].

[7] Microsoft Corporation: Creating a Business Logic Layer [online]. http://msdn.microsoft.com/en-us/library/aa581779(v=MSDN.10).aspx, 2010 [cit. 2010-05-01].

[8] Microsoft Corporation: Creating a Data Access Layer [online]. http://msdn.microsoft.com/en-us/library/aa581776(v=MSDN.10).aspx, 2010 [cit. 2010-05-01].

[9] Prokop, M.: Co je XHTML [online]. http://www.sovavsiti.cz/c01242.html, 2010-01-11 [cit. 2010-01-11].

[10] Prosise, J.: *Programování v Microsoft .NET*. Computer Press, 2003, 712 s., iSBN 80-7226-879-1.

[11] RISE to Bloome: Rise Editor [online]. http://www.risetobloome.com/PageItem.aspx?item=820, 2007-2009 [cit. 2009-01-11].

[12] Tomáš Herceg: Píšeme webovou aplikaci v ASP.NET krok za krokem [online]. http://www.vbnet.cz/clanek--105-zaciname_s_asp_net_dil_8_piseme_webovou_aplikaci_v_asp_net_krok_za_krokem_cast_1_.aspx, 2007 [cit. 2010-01-11].

[13] WWW stránky: Extensible Markup Language - Wikipedia [online]. http://cs.wikipedia.org/wiki/Extensible_Markup_Language, 2010 [cit. 2010-01-11].